

Робастное управление в режиме скольжения для роботизированных манипуляторов

ФИО студента: Южанин Андрей Алексеевич

Почта студента: andrei1998yuzhanin@gmail.com

GitHub: https://github.com/Juni0rResearcher/RobotControl_UR5e

Дата сдачи: 19.12.2025

Стек: Python 3.14, Mujoco

Задание:

Современные роботизированные манипуляторы часто сталкиваются с различными неопределенностями и возмущениями в процессе работы:

- Изменяющиеся массы полезной нагрузки
- Эффекты трения и демпфирования в суставах
- Неопределённости параметров модели
- Внешние возмущения

В то время как традиционное управление на основе обратной динамики показывает хорошие результаты при точном знании модели, управление в режиме скольжения (Sliding Mode Control, SMC) обеспечивает робастную производительность в присутствии как параметрических неопределённостей, так и ограниченных возмущений.

Динамика манипулятора с неопределенностями может быть выражена как:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + D\dot{q} + F_c(q) = t$$

где:

- $M(q)$: Неопределенная матрица масс
- $C(q, \dot{q})$: Неопределенные члены Кориолиса/центробежные силы
- $g(q)$: Неопределенный вектор гравитации
- D : Неизвестная диагональная матрица вязкого трения
- F : Неизвестный вектор кулоновского трения
- t : Управляющий вход

Задания:

1. [40 баллов] Контроллер обратной динамики

- Реализовать контроллер обратной динамики (Inverse Dynamics controller)

2 . [40 баллов] Контроллер в режиме скольжения

- Модифицировать модель робота UR5, включив:
- Дополнительную массу на конце эфектора
- Коэффициенты демпфирования суставов
- Кулоновское трение
- Реализовать контроллер в режиме скольжения
- Сравнить Inverse Dynamics и Sliding Mode

3 . [20 баллов] Реализация граничного слоя

- Проанализировать явление чаттеринга (дребезга):
- Причины и практические последствия
- Модификация с граничным слоем для сглаживания
- Оценить производительность при различных толщинах

граничного слоя Φ

- Проанализировать компромисс между робастностью и чаттерингом

Задание 1. Контроллер обратной динамики

Реализуем контроллер обратной динамики для робота UR5e (рисунок 1).

Зададим эталонные значения $k_p, k_d = 100, 20$

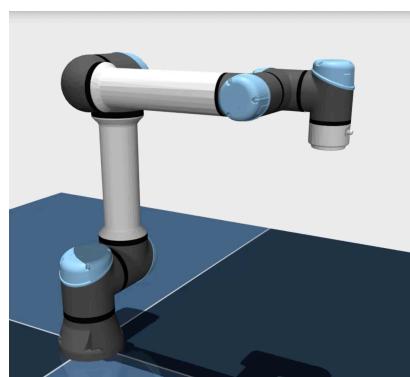


Рисунок 1 - Робот манипулятор UR5e

Для вычисления обратной динамики для начала зададим траекторию движения робота с плавным стартом.

```
q_offset = np.array([0.0, -1.57, 1.57, -1.57, -1.57, 0.0])

amplitude = np.array([0.15, 0.20, 0.15, 0.10, 0.10, 0.08])
frequency = np.array([0.2, 0.25, 0.3, 0.35, 0.4, 0.45])
phase = np.array([0.0, np.pi/4, np.pi/2, np.pi, 3*np.pi/4, np.pi/3])

warm_up_time = 0.5

if t < warm_up_time:
    # Держим текущую позу – плавный старт
    q_des = q.copy()
    dq_des = np.zeros(6)
    ddq_des = np.zeros(6)
else:
    # Траектория начинается с t = 0 в момент t = warm_up_time
    t_eff = t - warm_up_time

omega = 2 * np.pi * frequency
q_sin = amplitude * np.sin(omega * t_eff + phase)
q_des = q_offset + q_sin
dq_des = amplitude * omega * np.cos(omega * t_eff + phase)
ddq_des = -amplitude * omega**2 * np.sin(omega * t_eff + phase)
```

Контроллер обратной динамики с помощью Pinocchio реализуем следующим образом:

```
# Ошибки отслеживания
e = q_des - q
de = dq_des - dq

# Команда ускорения: feedforward + PD
qdd_cmd = ddq_des + KD * de + KP * e

# Номинальная динамика из Pinocchio
pin.computeAllTerms(pin_model, pin_data, q, dq)
M_hat = pin_data.M
nle_hat = pin_data.nle

# Управление
tau = M_hat @ qdd_cmd + nle_hat
```

Выведем на график работу контроллера (рисунок 2). На зеленом графике представлены ошибки положения. Для них характерен небольшой начальный пик (у некоторых шарниров чуть больше, у других меньше), затем быстро стремятся к нулю и остаются в очень узком диапазоне вокруг нуля. По графикам видно, что стационарная ошибка практически отсутствует, а максимальные пиковые ошибки по положению порядка десятых–сотых радиана, что для траекторного управления манипулятором обычно считается хорошим результатом, если такие значения удовлетворяют требованиям задачи.

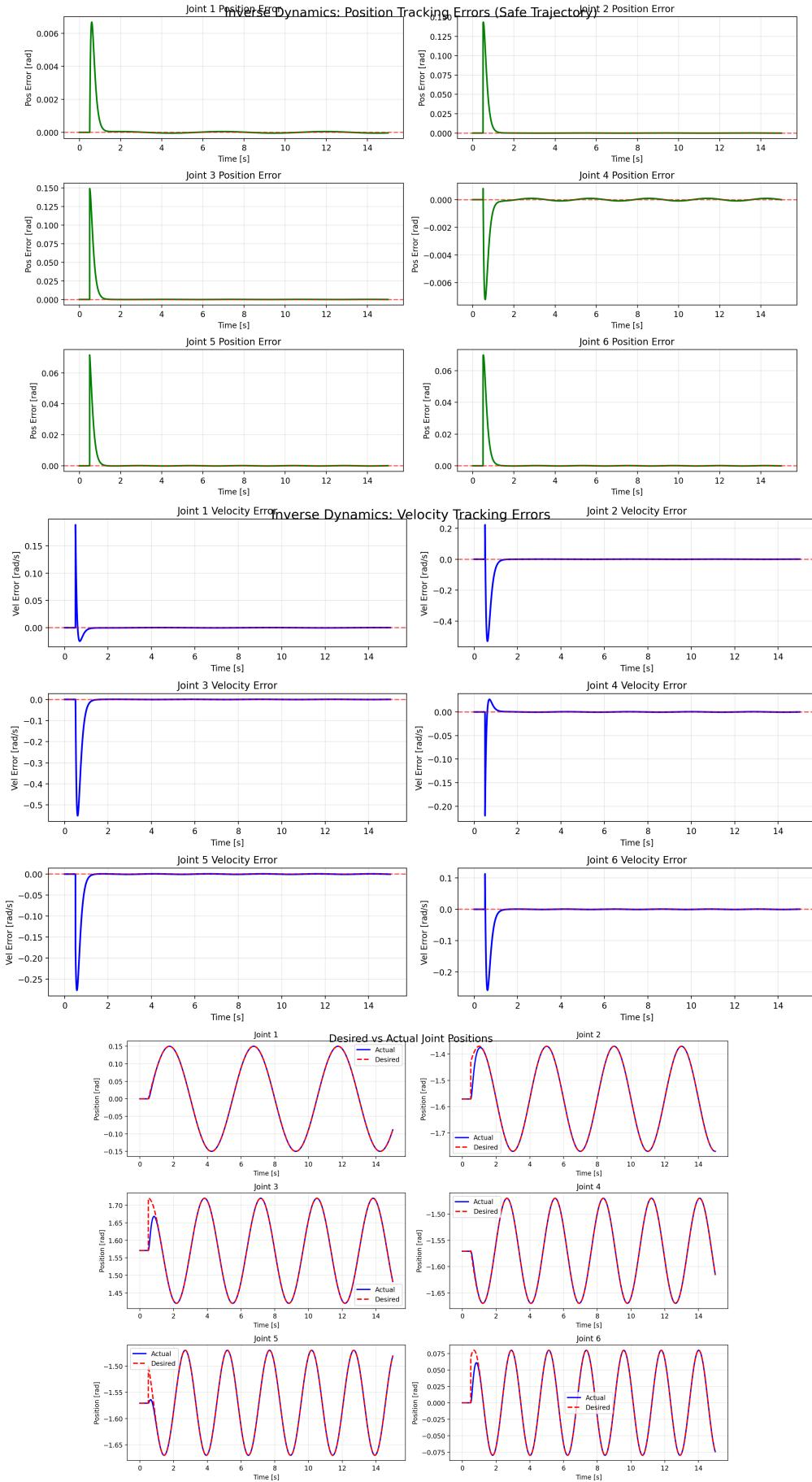


Рисунок 2 - Визуализация работы контроллера на данных

Проанализируем графики скорости (синий график). Ошибки скорости у всех 6 суставов имеют кратковременный бросок в начале, после чего экспоненциально затухают к нулю и остаются практически на нуле на всём интервале. Можно сказать, что PD-звено в обратной динамике успешно настроено, так как есть переходный процесс, но система не уходит в устойчивые колебания и не показывает систематического смещения.

На графике сравнения ожидаемой и действительной траектории можно наблюдать скачок в начале работы роботы, а затем выравнивание и полное соответствие ожиданию.

Для полноту картины выведем графики моментов для каждого шарнира, что понимать их усилие в зависимости от времени (рисунок 3).

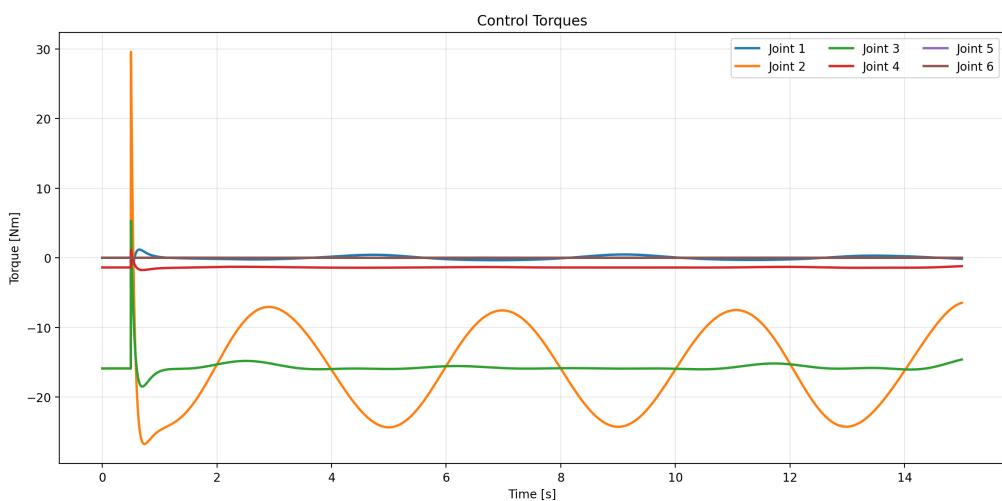


Рисунок 3 - Графики моментов шарниров в зависимости от времени

На графике крутящих моментов видно резкие пики в начале движения (особенно у 2-го и 3-го шарнира), затем все моменты быстро переходят в плавный режим с малыми колебаниями.

Такие всплески обычно связаны с разрывом начальных условий (старт не из равновесия траектории), жёсткой траекторией в начале или достаточно высокими коэффициентами обратной связи. Обычно для ограничения крутящего момента принимают 150 Н*м, по нашему графику мы наблюдаем, что

полученное значение не превышает $30 \text{ Н}^*\text{м}$, что находится в пределах нормы, излишней нагрузки на шарниры нет.

Реализованный контроллер обратной динамики эффективно компенсирует нелинейную динамику робота и обеспечивает хорошее следование траектории.

Проведем исследование, как влияет добавление массы на end-effector и установка коэффициентов демпфирования и трения шарниров.

```
#Установка коэффициентов демпфирования суставов
damping = np.array([0.5, 0.5, 0.5, 0.1, 0.1, 0.1]) # Н/(рад/с)
sim.set_joint_damping(damping)
# Установка коэффициентов трения суставов
friction = np.array([1.5, 0.5, 0.5, 0.1, 0.1, 0.1]) # НМ
sim.set_joint_friction(friction)
# Модификация массы конца эфектора
sim.modify_body_properties("end_effector" , mass=4)
```

Визуализируем работу контроллера обратной динамики с добавлением условий, возмущающих систему. На рисунке 4 представлен график ошибки положения для каждого шарнира.

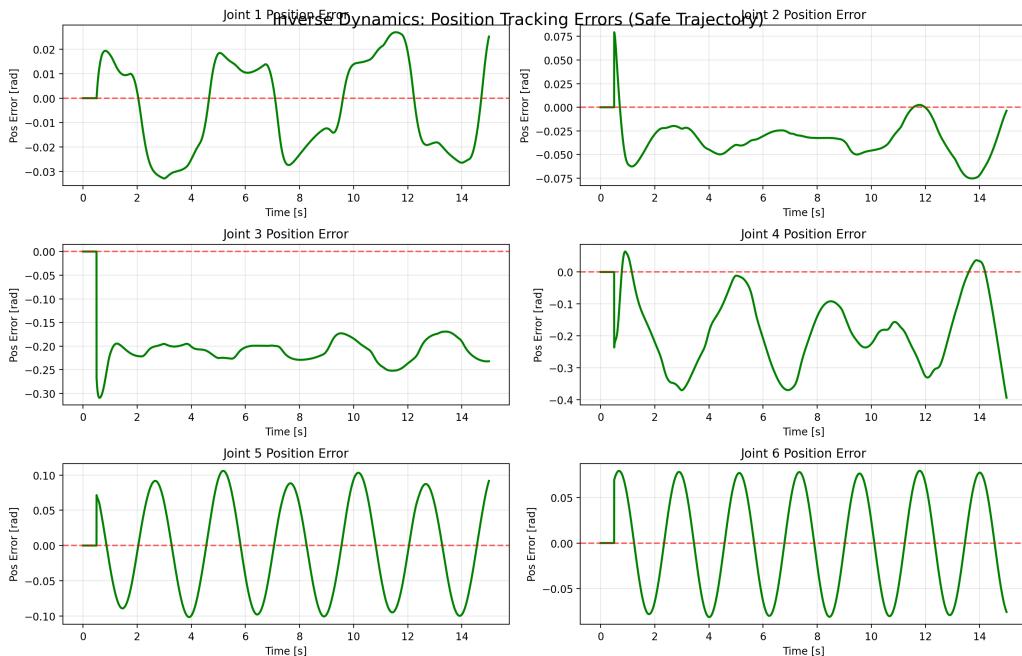


Рисунок 4 - Графики ошибки положения для шарниров в зависимости от времени после добавления дополнительных условий в систему

Таким образом, если раньше в идеальных условиях мы наблюдали ошибку только на старте, то теперь видим отклонение на протяжении всей работы робота манипулятора. Правильное положение не достигается. Проанализируем графики скорости суставов (рисунок 5).

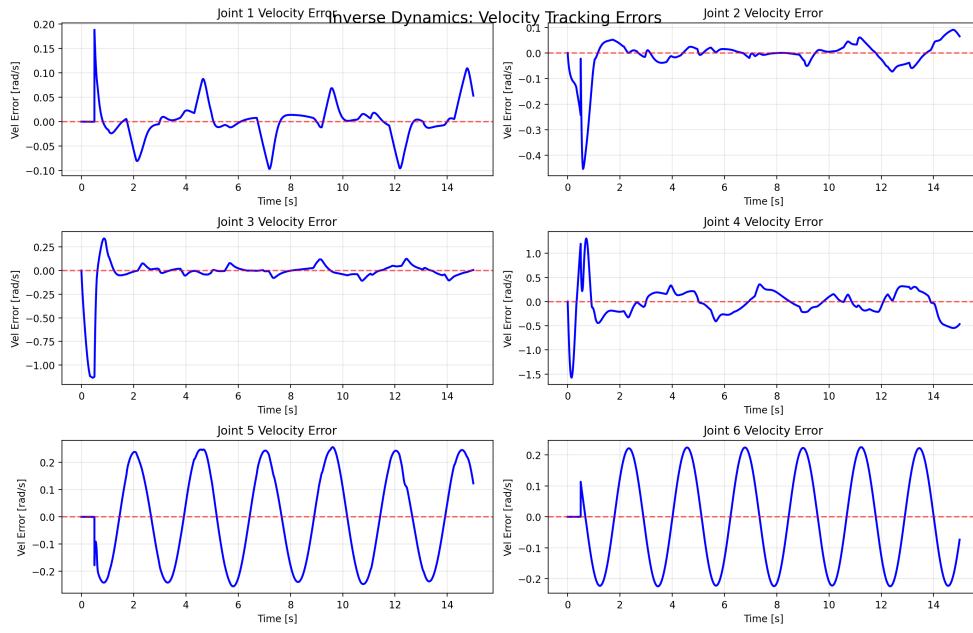
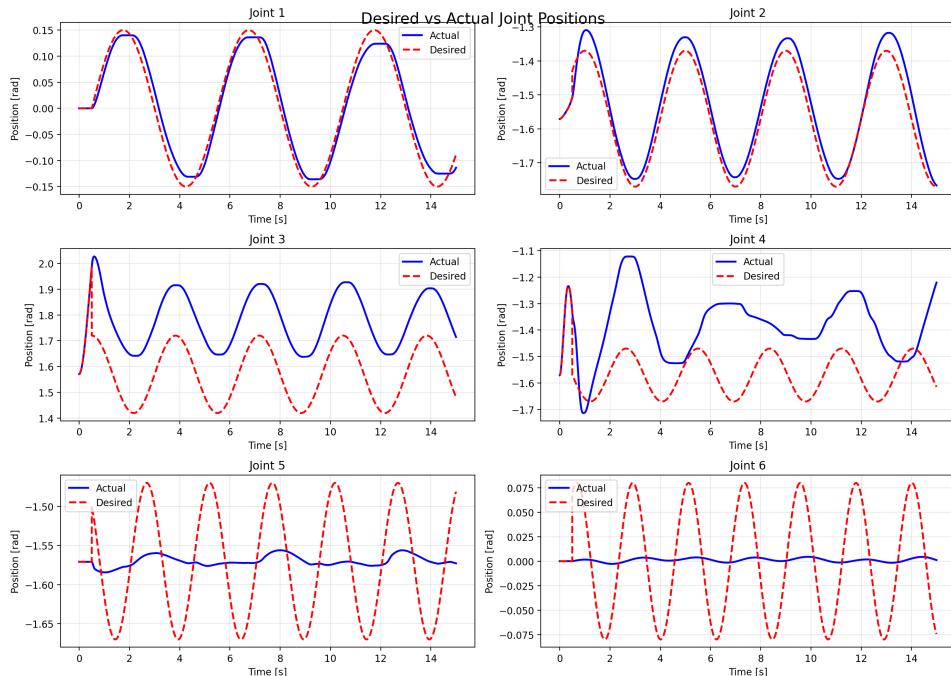


Рисунок 5 - Графики скорости суставов в зависимости от времени после добавления дополнительных условий в систему

По графике скорости тоже наблюдаем постоянное изменение скорости, система не приходит в стабильное состояние, система постоянно пытается найти баланс. На рисунке 6 представлен график сравнения ожидаемого положения шарниров и их фактического расположения в пространстве.



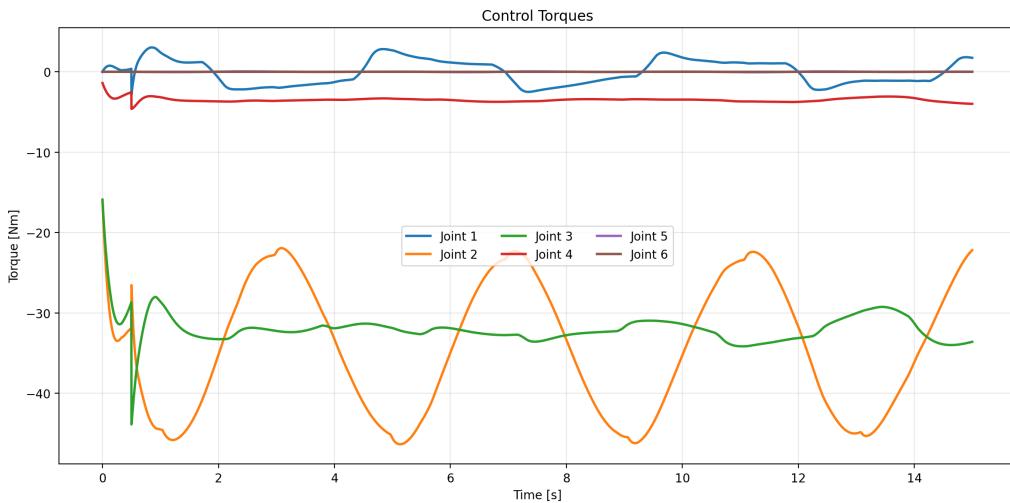


Рисунок 6 - Сравнение ожидаемого и фактического положения суставов в пространстве в зависимости от времени после добавления дополнительных условий в систему

Рисунок 7 - График крутящего момента шарниров в момент времени после добавления дополнительных условий в систему

Проанализируем поведение моментов (рисунок 7). На графике наблюдаем, что в начале временного промежутка присутствует скачок момента, так как робот ловит баланс, а затем неровная работа на протяжении всего времени.

В таблице 1 представлено сравнение RMSE позиции и скорости в идеальных условиях и условиях с возмущением системы.

Таблица 1 - сравнение метрик RMSE позиции и скорости шарниров

	Контроллер обратной динамики в идеальных условиях		Контроллер обратной динамики в условиях добавления массы и коэффициентов трения и демпфирования шарниров	
Шарнир	RMSE Position Errors (Rad)	RMSE Velocity Errors (rad/s)	RMSE Position Errors (Rad)	RMSE Velocity Errors (rad/s)
Joint 1	0.001787	0.018787	0.020258	0.037935
Joint 2	0.033207	0.141891	0.038129	0.109982
Joint 3	0.032857	0.147256	0.204040	0.307839
Joint 4	0.001934	0.021950	0.216401	0.446363
Joint 5	0.014270	0.072918	0.060479	0.169048
Joint 6	0.016233	0.069274	0.052852	0.149805

Таким образом видим значительное увеличение ошибки RMSE как для позиции, так и для скорости, что говорит о нестабильности системы.

Так как регулятор использует номинальную матрицу M_{hat} из Pinocchio, а реальная масса увеличилась на 4 кг, то мы наблюдаем систематическую ошибку при скорости и положениях. Быстрые движения могут привести к падению робота.

Задание 2. Контроллер в режиме скольжения.

Реализуем контроллер в режиме скольжения. Контроллер в режиме скольжения (sliding mode controller, SMC) - это нелинейный робастный регулятор, который заставляет состояние системы «прилипнуть» к специально выбранной поверхности в пространстве состояний и дальше двигаться по ней, несмотря на возмущения и неопределенность модели.

Зададим скользящую поверхность:

$$s = e^{\cdot} + \lambda e, \text{ где:}$$

$$e = q_d - q$$

$$e^{\cdot} = \dot{q}_d - \dot{q}$$

В коде реализуем так:

```
e = q_des - q
de = dq_des - dq

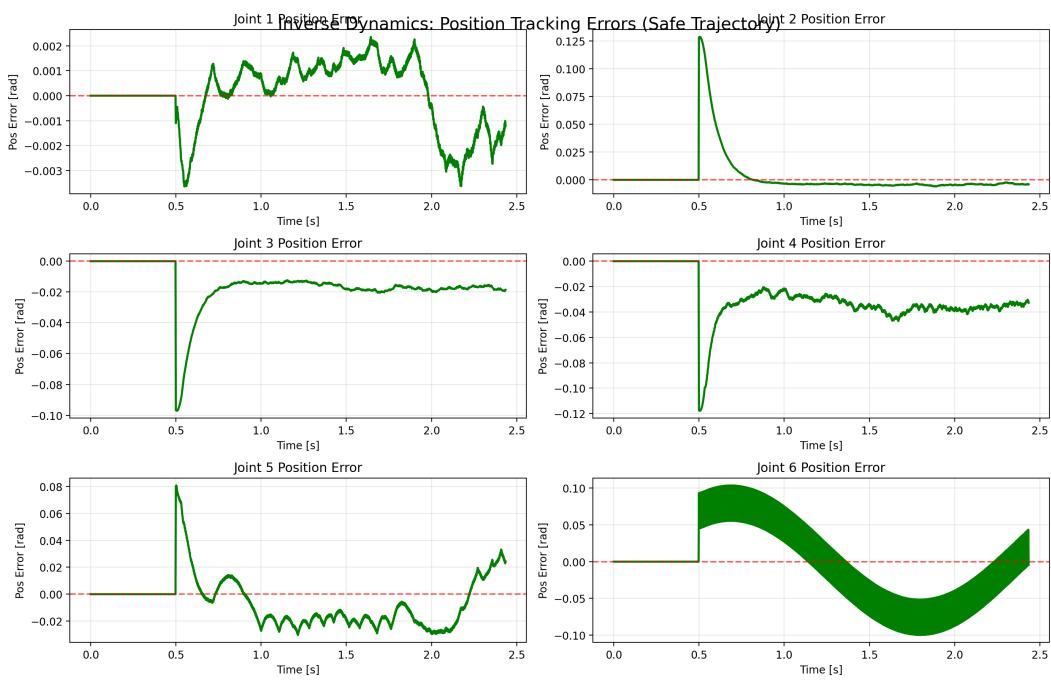
# Поверхность скольжения
s = de + Lambda * e

# Номинальная часть (эквивалентный контроль)
pin.computeAllTerms(pin_model, pin_data, q, dq)
M_hat = pin_data.M
nle_hat = pin_data.nle

# Эквивалентное управление: компенсирует номинальную динамику
v = ddq_des + Lambda * de
tau_nominal = M_hat @ v + nle_hat
tau_robust = K_robust_vec * np.sign(s)
if t < 0.5:
    tau_robust *= (t / 0.5)
tau = tau_nominal + tau_robust
```

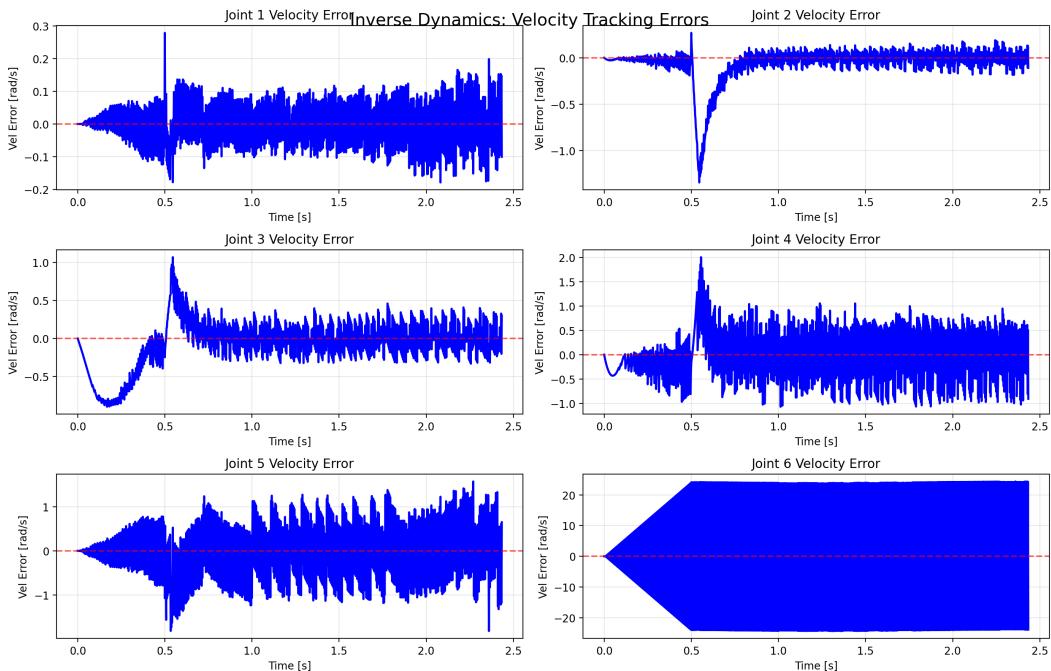
Коэффициент K и λ зададим так:

```
K_robust_vec = np.array([20.0, 50.0, 30.0, 10.0, 10.0, 8.0])
Lambda = 12.0
```

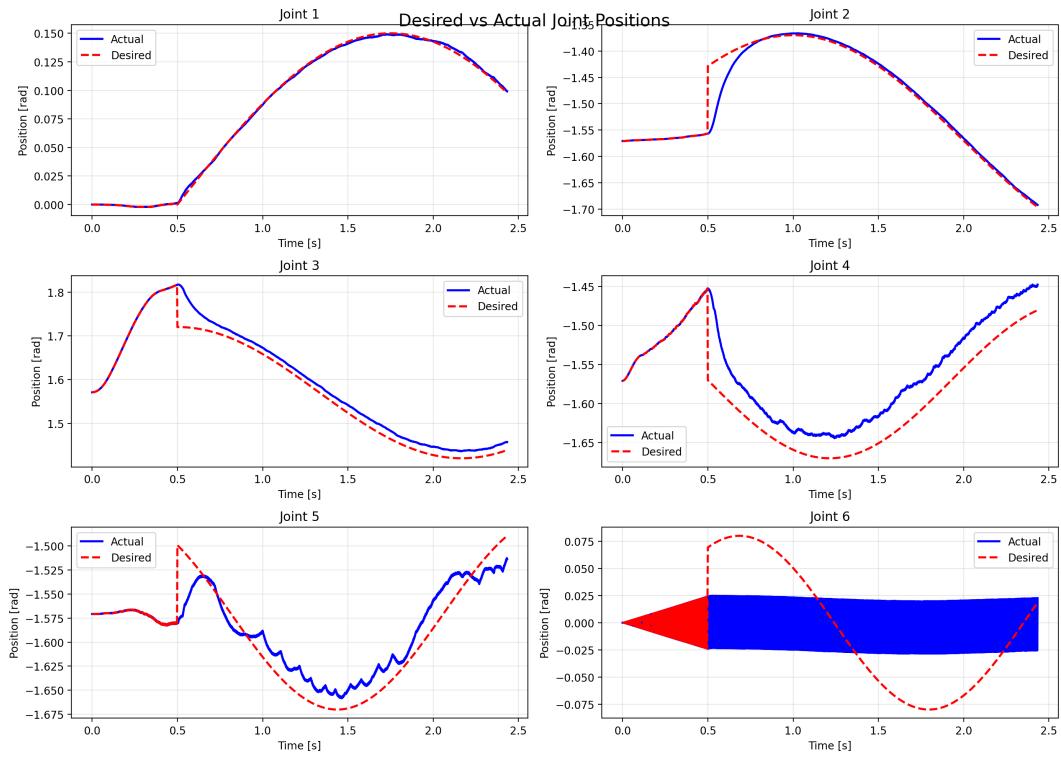


На рисунке 8 представлены графики позиции ошибки для каждого сустава в зависимости от времени. График показывает, как системы пытается прийти в идеальное состояние, графики не такие скачкообразные, как были у контроллера обратной динамики.

Рисунок 8 - График ошибки положения для контроллера в режиме



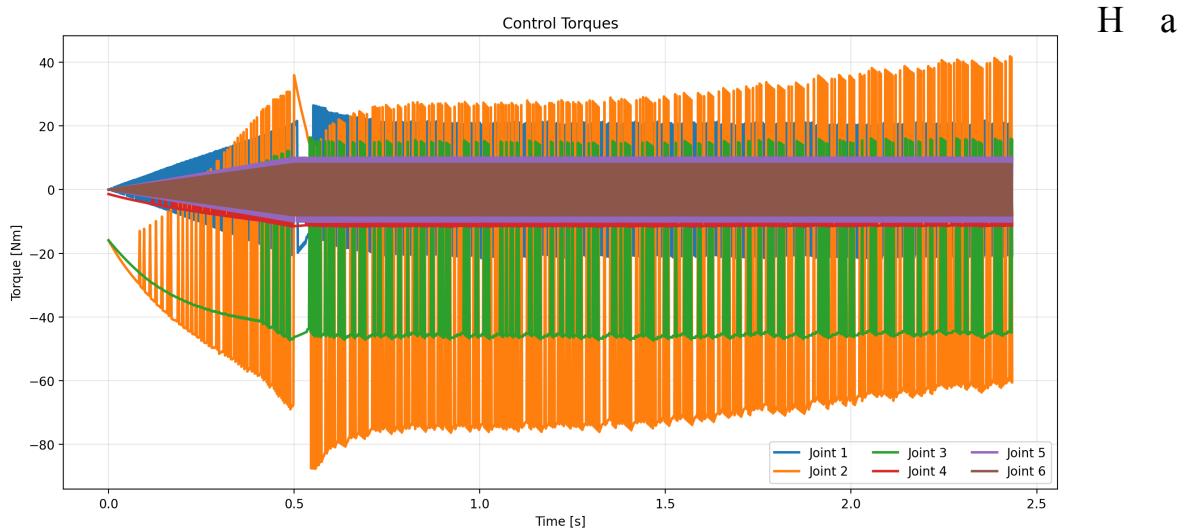
СКОЛЬЖЕНИЯ



Рассмотрим графики ошибки скорости (рисунок 9). Видим, что данные стали более шумными и увеличилась амплитуда колебаний. Управление регулирует скорость с большей частотой для поддержания заданной траектории.

Рисунок 9 - График ошибки скорости для контроллера в режиме скольжения

Рисунок 10 - Сравнение ожидаемого положения и фактического для контроллера в режиме скольжения



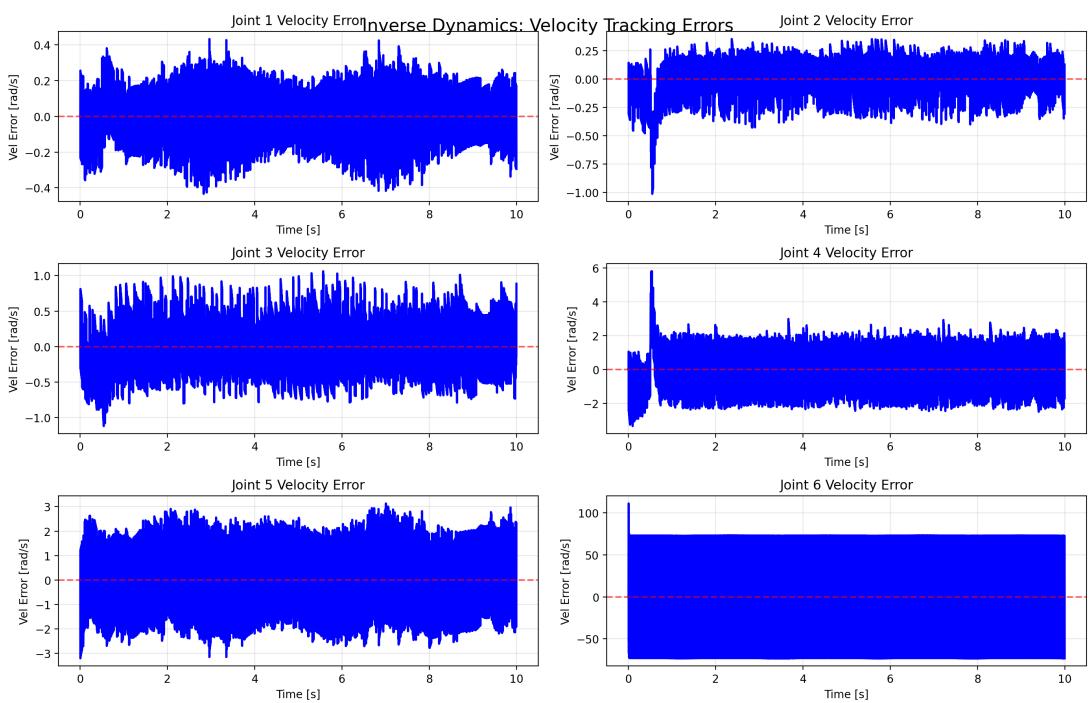
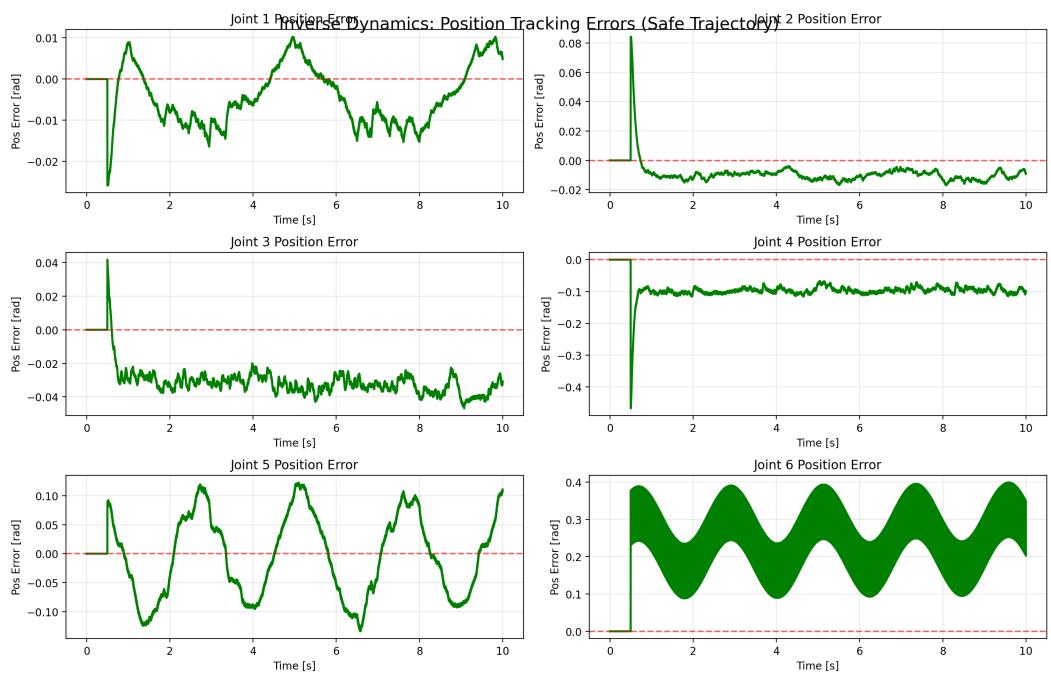
на рисунке 10 представлен сравнительный график фактического положения шарниров робота манипулятора с ожидаемым. По графику видим, что в

траекторию идеально попадает первый шарнир, второй (с учетом сглаживания резкого перемещения) и 3 почти удалось отрегулировать. Для остальных видим большое отклонение.

Рисунок 11 - Крутящие моменты шарниров в моменты времени для контроллера в режиме скольжения

По крутящему моменты видим, что он значительно увеличился для 2 и 3 суставов, но все еще находится в границах 150 Н*м. Видим, что момент регулируется не плавно, присутствуют шумы.

Проблема нашего регулятора заключается в случайному подборе λ и K , которые в текущей заданной траектории обеспечивают стабильность работы системы, но в случае изменения конфигурация, она не гарантирована. Определим эти параметры методом Ляпунова. В коде добавим:



```
# 1. Оценка верхней границы возмущений delta_max от неопределённостей
# Лишняя масса: 4 кг на конце эфектора
m_payload = 4.0
g = 9.81
L_arm = 0.9
```

В результате получаем $\lambda = 10.0$, $K = [32.1 \ 79.5 \ 65.4 \ 27.7 \ 27.7 \ 24.1]$

Сравним графики ошибки скорости, положения , а также крутящие моменты шарниров и траекторию. Рассмотрим рисунок 12 с графиком ошибки положения.

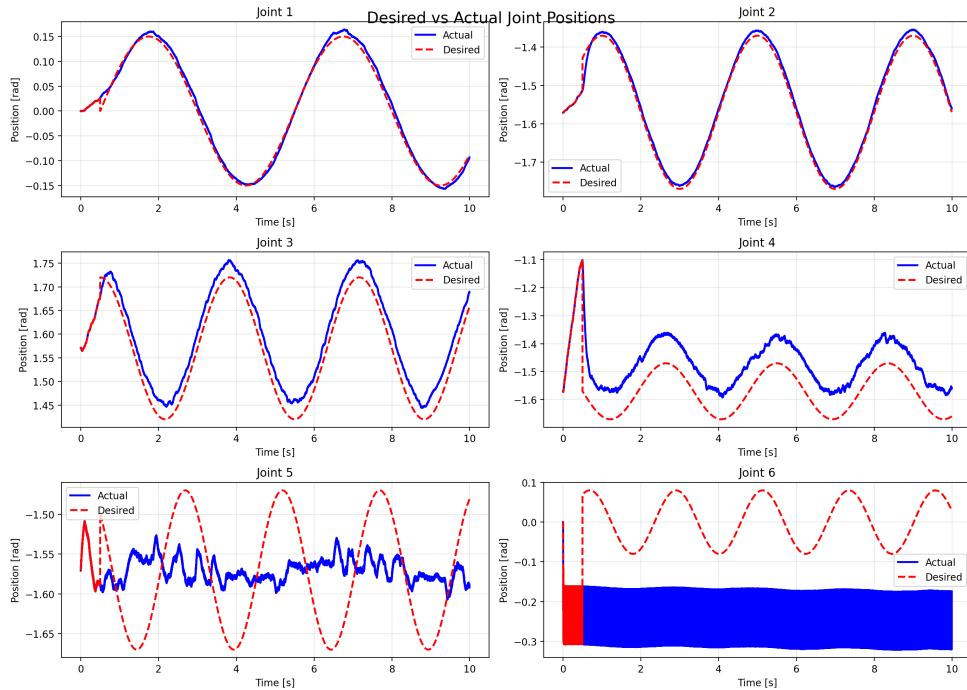


Рисунок 12 - График ошибки положения для контроллера в режиме скольжения с рассчитанными K и λ

Теперь на графике видим, что для 1 и 3 сустава отклонение увеличилось, но в целом шарниры ведут себя стабильнее, относительно предыдущего подбора параметров.

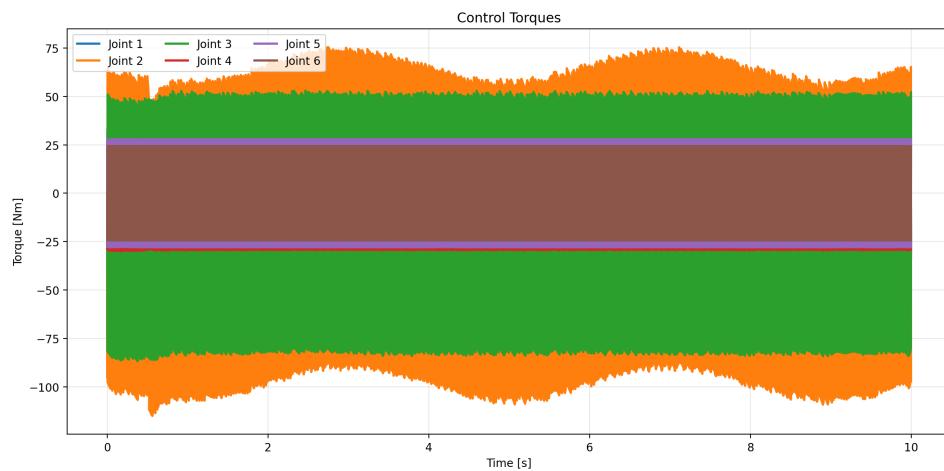


Рисунок 13 - График ошибки скорости для контроллера в режиме скольжения с рассчитанными K и lambda

На рисунке 13 можем наблюдать увеличение ошибки по скорости, что вызвано новым коэффициентами регулирования K для каждого шарнира.

Сравним ожидаемую и фактическую траекторию движения шарниров на рисунке 14.

Рисунок 14 - Сравнение ожидаемой и фактической траектории шарниров для контроллера в режиме скольжения с рассчитанными K и lambda

Подобранные методом Ляпунова параметры lambda и K, привели к увеличению ошибки для 5 и 6 шарниров, что говорит о недостаточном регулировании системы. На рисунке 15 представлен график моментов.

Рисунок 15 - График моментов суставов

График моментов показывает смещение их в отрицательную сторону, теперь нужно прикладывать еще больше усилий, для сохранения равновесия.

В таблице 2 представлено сравнение RMSE при различных параметрах K и Lambda. По ошибкам наблюдается тенденция к их увеличению, но не смотря на это система становится адаптивной к разным траекториям, а также графики более плавные.

Таблица 2 - сравнение RMSE для постоянных K и lambda с подбором по Ляпунову

	Постоянные K и Lambda		Подбор K и lambda по Ляпунову	
Шарнир	RMSE Position Errors (Rad)	RMSE Velocity Errors (rad/s)	RMSE Position Errors (Rad)	RMSE Velocity Errors (rad/s)
Joint 1	0.001364	0.079486	0.008171	0.211072
Joint 2	0.020999	0.206752	0.012099	0.194371
Joint 3	0.021909	0.329380	0.031752	0.466227
Joint 4	0.034982	0.558089	0.099665	1.573185
Joint 5	0.019529	0.753079	0.069627	1.982136
Joint 6	0.054135	22.290334	0.256652	72.901170

В таблице 3 представлен сравнительный анализ контроллера обратной динамики с возмущением системы с контроллером в режиме скольжения. Несмотря на увеличение RMSE, нужно проанализировать графики. По графики мы увидим большую сходимость системы для контроллера в режиме скольжения, что обеспечивает лучшую стабильность работы.

Таблица 3 - Сравнительный анализ RMSE между контроллерами

	Контроллер обратной динамики		Подбор K и lambda по Ляпунову	
Шарнир	RMSE Position Errors (Rad)	RMSE Velocity Errors (rad/s)	RMSE Position Errors (Rad)	RMSE Velocity Errors (rad/s)
Joint 1	0.020258	0.037935	0.008171	0.211072
Joint 2	0.038129	0.109982	0.012099	0.194371
Joint 3	0.204040	0.307839	0.031752	0.466227
Joint 4	0.216401	0.446363	0.099665	1.573185
Joint 5	0.060479	0.169048	0.069627	1.982136
Joint 6	0.052852	0.149805	0.256652	72.901170

Отличия между подходами заключаются в следующем:

1. Обратная динамика:

- предполагает, что мы обладаем всей информацией о роботе, знаем его идеально и компенсируем динамику с помощью ПД регулятора:

$$\tau = M(q) \cdot \ddot{q}_{des} + C(q, \dot{q}) \cdot \dot{q} + g(q) + PD$$

- Обеспечивается локальная устойчивость
- Экспоненциальная сходимость
- Точная модель
- Простота настройки

2. Sliding Mode Control:

- предполагается, что мы не знаем робота точна, заставляем его следовать динамике силой:

$$\tau = \hat{M}(\ddot{\hat{q}}_{des} + \lambda \dot{e}) + \hat{C} \cdot \dot{\hat{q}} + \hat{g} - K \cdot sign(s),$$

$$s = e^{\cdot} + \lambda e$$

- Обеспечивается глобальная устойчивость
- Сходимость за определенное время

- Ограниченные возмущения
- Сложность настройку в силу появления lambda и K

Главный недостаток метода SMC это чаттеринг нежелательные высокочастотные колебания управления в системах с разрывными законами управления, таких как скользящий режим (SMC). Возникает при использовании функции sign(s). На полученных графиках и видеозаписи явно прослеживается этот эффект.

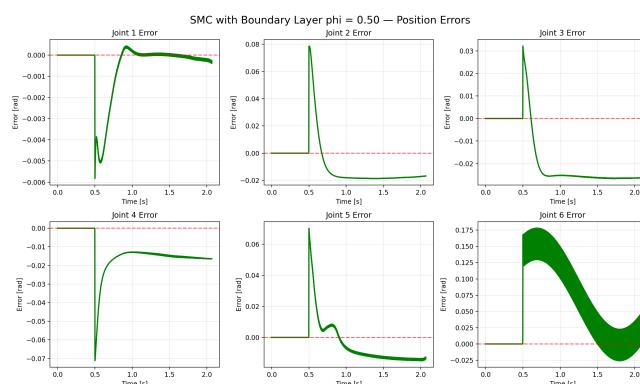
1. Как возникает: Система достигает скользящей поверхности $s = 0$
2. Контроллер включает максимальное управление $+K$
3. Система "проскаакивает" поверхность
4. Контроллер переключается на $-K$. Процесс повторяется с высокой частотой

Результат: Бесконечная последовательность переключений K при $s = 0$

Это приводит к повышенному износу шарниров (страдают подшипники, редуктор, приводы из-за перегрева). Снижается общее КПД системы, так как происходит постоянное переключение. Для борьбы с чаттерингом используют Boundary Layer (Границный слой) (Задание 3).

Задание 3. Границный слой (Boundary Layer)

Как сказано ранее гранитный слой позволяет бороться с чаттерингом, снижая нагрузку на шарниры. Реализуем функцию управления



```

def controller(q: np.ndarray, dq: np.ndarray, t: float) -> np.ndarray:
    q_des, dq_des, ddq_des = generate_trajectory(q, t)

    e = q_des - q
    de = dq_des - dq
    s = de + Lambda * e

    pin.computeAllTerms(pin_model, pin_data, q, dq)
    M_hat = pin_data.M
    nle_hat = pin_data.nle

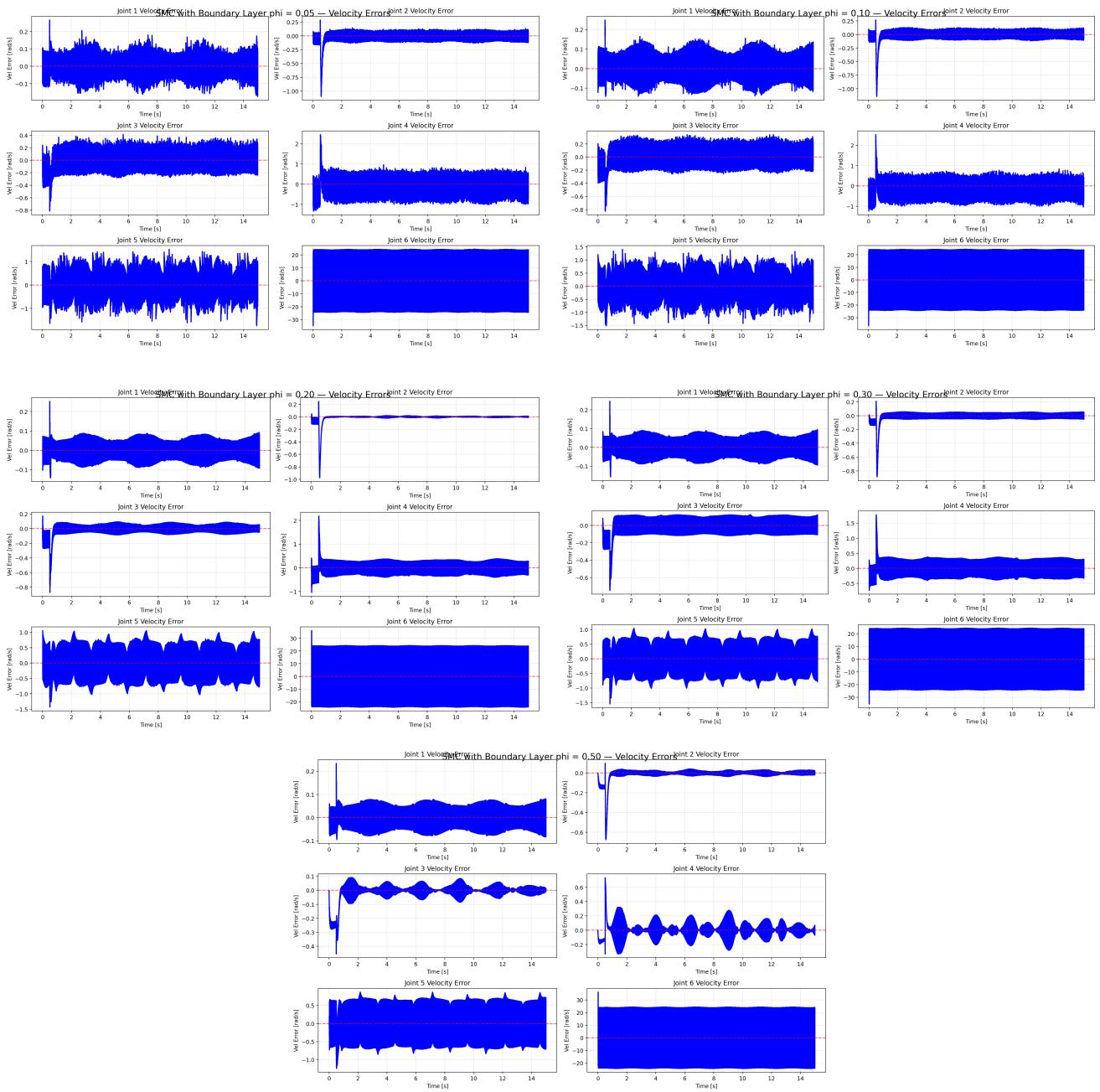
    tau_nominal = M_hat @ (ddq_des + Lambda * de) + nle_hat
    tau_robust = K_robust_vec * np.clip(s / phi, -1.0, 1.0)
    tau = tau_nominal + tau_robust
    tau = np.clip(tau, -150, 150)

    # Логирование
    simulation_data['time'].append(t)
    simulation_data['positions'].append(q.copy())
    simulation_data['velocities'].append(dq.copy())
    simulation_data['torques'].append(tau.copy())
    simulation_data['desired_positions'].append(q_des.copy())
    simulation_data['desired_velocities'].append(dq_des.copy())
    simulation_data['sliding_surface'].append(s.copy())

    return tau
    return controller

```

Ключевое действие - замена $\text{sign}(s)$ на $\text{np.clip}(s / \phi, -1.0, 1.0)$. Таким



образом, появляется гиперпараметр ϕ , который нужно подбирать. Он определяет толщину пограничного слоя. Запустим контроллер для робота разными Φ и проанализируем результат. На рисунке 16 представлен график ошибки положения при разных $\Phi = [0.05, 0.1, 0.2, 0.3, 0.5]$.

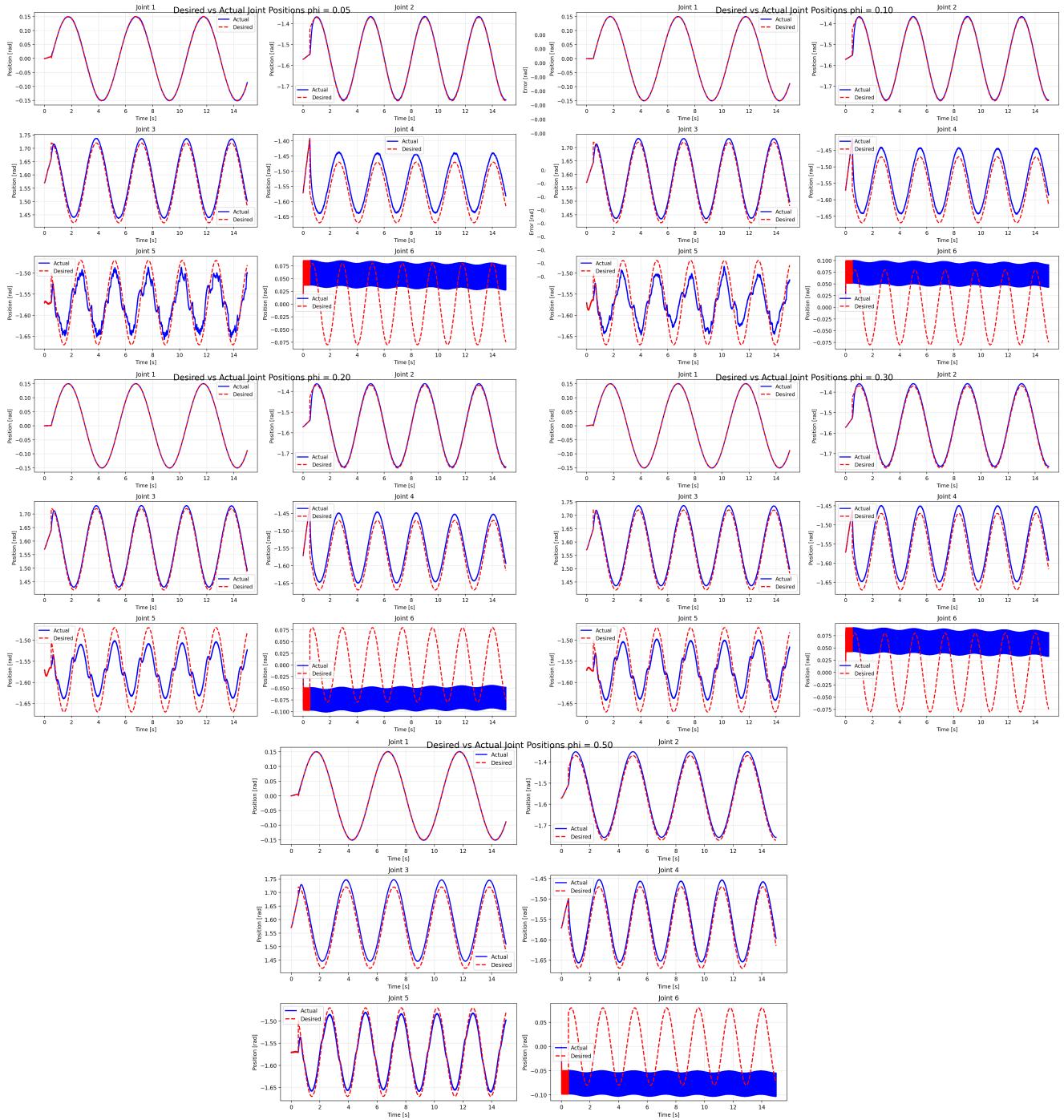
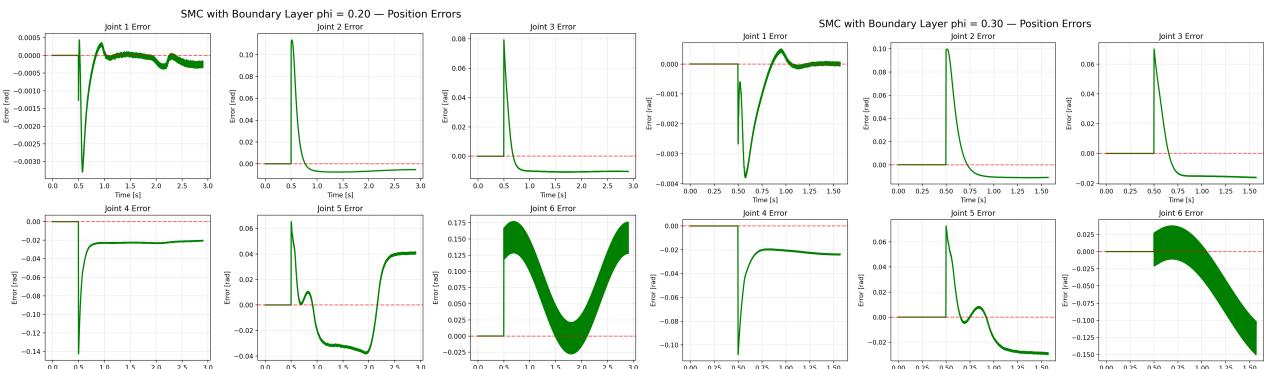


Рисунок 15 - Сравнение ошибки положения при разных ϕ

По графику видим, что чаттеринг снижается по мере увеличения ϕ , но это влияет на ошибку. Самый лучший график для $\phi = 0.3$. Видно, что здесь



меньше ошибка для 3 и 4 шарниров, а также меньше чаттеринг у 6 шарнира. Проанализируем поведение скорости (рисунок 16).

По графику скорости видим, что амплитуда скорости для всех суставов существенно снизилась, что свидетельствует об уменьшении чаттеринга и добавлении стабильности в системе.

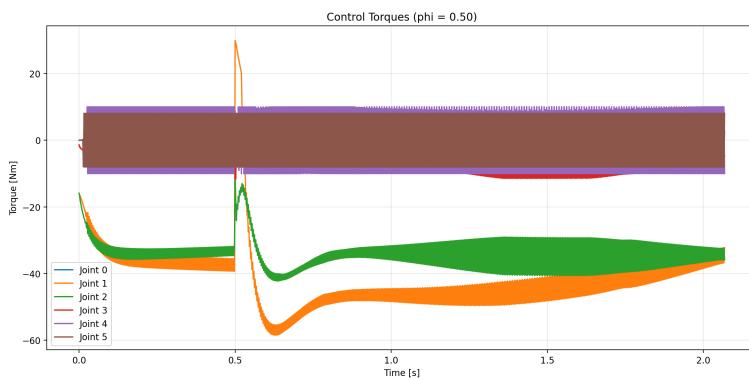
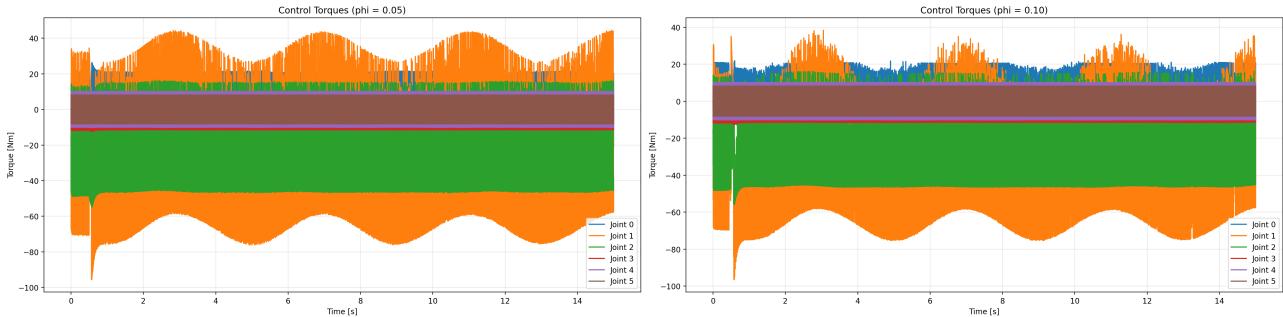


Рисунок 16 - Сравнение ошибки положения при разных ϕ

Сравним фактическую траекторию движения с ожидаемой (рисунок 17). По графикам видно, что добавление Boundary Layer привело к стабилизации траектории для всех шарниров. Лучше всего система себя показывает при значении $\phi = 0.5$, реальная синусоида почти повторяет ожидаемую.

Рисунок 17 - Сравнение ожидаемой траектории с реальной для SMC с Boundary Layer при различных ϕ

Построим графики поведения моментов (рисунок 18). График момента показывает, что с увеличением ϕ чаттеринг уменьшается и система начинает



работать стабильнее. Например, для 1 и 2 шарниров явно прослеживается стабилизация и снижение чаттеринга.

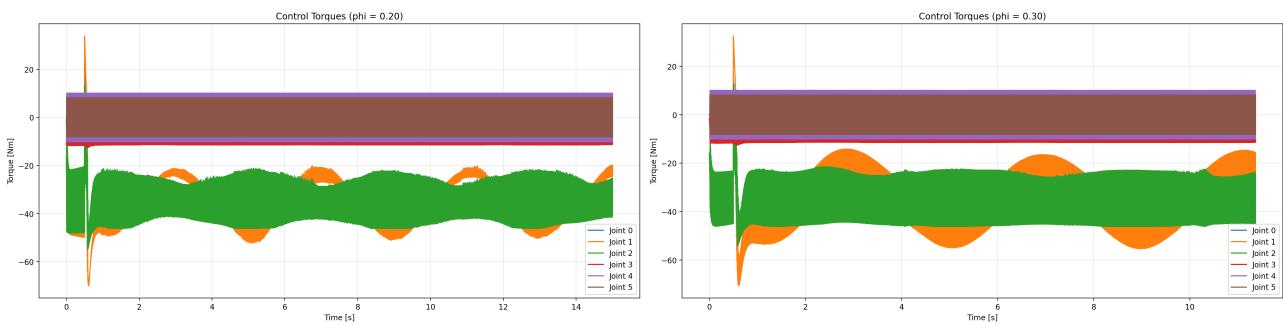


Рисунок 18 - Сравнение моментов для SMC с Boundary Layer при различных phi

Проанализируем RMSE позиции для разных phi в таблице 4

Таблица 4 - Анализ влияния phi на RMSE позиции шарниров

Шарнир	RMSE Position Errors (Rad)				
	Phi = 0.05	Phi = 0.1	Phi = 0.2	Phi = 0.3	Phi = 0.5
Joint 1	0.001555	0.000278	0.000304	0.000427	0.000692
Joint 2	0.009466	0.010009	0.010327	0.011663	0.016522
Joint 3	0.017428	0.014830	0.011161	0.015923	0.025595
Joint 4	0.032686	0.028095	0.022606	0.021058	0.015585
Joint 5	0.026933	0.032298	0.032286	0.027549	0.012470
Joint 6	0.080074	0.090546	0.093804	0.083626	0.096915

В таблице 5 представлен сравнительный анализ ошибки скоростей для различных значений phi.

Таблица 5 - Влияние phi на ошибку RMSE скоростей шарниров

Шарнир	RMSE Velocity Errors [rad/s]:				
	Phi = 0.05	Phi = 0.1	Phi = 0.2	Phi = 0.3	Phi = 0.5
Joint 1	0.088597	0.084490	0.064049	0.067371	0.060769
Joint 2	0.100003	0.096845	0.070923	0.078644	0.062917
Joint 3	0.173698	0.158332	0.087249	0.120396	0.065748
Joint 4	0.539679	0.481113	0.328344	0.333150	0.126325
Joint 5	0.797836	0.764281	0.661342	0.678389	0.640405
Joint 6	23.989585	24.011540	24.022384	24.023745	24.020531

Таким образом в результате анализа всех полученных графиков и RMSE положения и скорости, можно сделать вывод о том, что оптимальным значением $\phi = 0,5$, это обеспечивает лучшее схождение и стабилизацию системы.